# Explanation of teamperf.py Code

The teamperf.py script analyzes team performance based on historical IPL match data and seasonal trends. The script performs the following key steps:  
  
1. Import Libraries:  
 import pandas as pd  
 import matplotlib.pyplot as plt  
 import seaborn as sns  
 - The script imports the necessary libraries: pandas for data manipulation, matplotlib.pyplot for plotting, and seaborn for creating attractive and informative statistical graphics.  
  
2. Load the Data:  
 deliveries\_df = pd.read\_csv('C:/Users/SHEEYASH/Documents/GitHub/IPL-analysis-/Team1/shreeyash/deliveries.csv')  
 matches\_df = pd.read\_csv('C:/Users/SHEEYASH/Documents/GitHub/IPL-analysis-/Team1/shreeyash/matches.csv')  
 - The script reads the deliveries.csv and matches.csv files into pandas DataFrames named deliveries\_df and matches\_df, respectively.  
  
3. Extract Relevant Columns from matches\_df:  
 matches\_df = matches\_df[['id', 'season', 'team1', 'team2', 'winner']]  
 - Only the necessary columns (id, season, team1, team2, winner) are extracted from the matches\_df DataFrame for analysis.  
  
4. Calculate Total Runs Scored by Each Team:  
 total\_runs = deliveries\_df.groupby('batting\_team')['total\_runs'].sum().reset\_index()  
 - The script groups the deliveries\_df DataFrame by batting\_team and calculates the total runs scored by each team. The result is stored in a DataFrame named total\_runs.  
  
5. Calculate Total Wickets Taken by Each Team:  
 wickets = deliveries\_df[deliveries\_df['is\_wicket'] == 1].groupby('bowling\_team').size().reset\_index(name='wickets')  
 - The script filters the deliveries\_df DataFrame to include only the rows where a wicket was taken (is\_wicket == 1). It then groups the DataFrame by bowling\_team and counts the total number of wickets taken by each team. The result is stored in a DataFrame named wickets.  
  
6. Calculate Win/Loss Ratio for Each Team:  
 wins = matches\_df.groupby('winner').size().reset\_index(name='wins')  
 matches\_played = pd.melt(matches\_df, id\_vars=['id', 'season'], value\_vars=['team1', 'team2'], var\_name='team\_type', value\_name='team')  
 matches\_played = matches\_played.groupby('team').size().reset\_index(name='matches\_played')  
 win\_loss\_ratio = pd.merge(wins, matches\_played, left\_on='winner', right\_on='team')  
 win\_loss\_ratio['win\_loss\_ratio'] = win\_loss\_ratio['wins'] / win\_loss\_ratio['matches\_played']  
 - The script calculates the number of wins for each team by grouping the matches\_df DataFrame by winner.  
 - It then uses the pd.melt function to transform the matches\_df DataFrame so that each row represents a team that played in a match, along with the season and the match ID.  
 - The script groups the melted DataFrame by team to count the total number of matches played by each team.  
 - It merges the wins and matches played DataFrames to calculate the win/loss ratio for each team. The result is stored in a DataFrame named win\_loss\_ratio.  
  
7. Merge Performance Metrics into a Single DataFrame:  
 team\_performance = pd.merge(total\_runs, wickets, left\_on='batting\_team', right\_on='bowling\_team')  
 team\_performance = pd.merge(team\_performance, win\_loss\_ratio[['winner', 'win\_loss\_ratio']], left\_on='batting\_team', right\_on='winner')  
 team\_performance = team\_performance[['batting\_team', 'total\_runs', 'wickets', 'win\_loss\_ratio']]  
 - The script merges the total\_runs, wickets, and win\_loss\_ratio DataFrames into a single DataFrame named team\_performance. This DataFrame contains the performance metrics (total runs, total wickets, win/loss ratio) for each team.  
  
8. Analyze Seasonal Trends:  
 seasonal\_performance = matches\_df.groupby(['season', 'winner']).size().reset\_index(name='wins')  
 - The script groups the matches\_df DataFrame by season and winner to count the number of wins for each team in each season. The result is stored in a DataFrame named seasonal\_performance.  
  
9. Plot Total Runs Scored by Each Team:  
 plt.figure(figsize=(14, 8))  
 sns.barplot(x='total\_runs', y='batting\_team', data=team\_performance, palette='viridis')  
 plt.title('Total Runs Scored by Each Team')  
 plt.xlabel('Total Runs')  
 plt.ylabel('Team')  
 plt.show()  
 - The script creates a bar plot to visualize the total runs scored by each team. The sns.barplot function is used to create the plot, and the viridis palette is used for coloring.  
  
10. Plot Total Wickets Taken by Each Team:  
 plt.figure(figsize=(14, 8))  
 sns.barplot(x='wickets', y='bowling\_team', data=team\_performance, palette='magma')  
 plt.title('Total Wickets Taken by Each Team')  
 plt.xlabel('Total Wickets')  
 plt.ylabel('Team')  
 plt.show()  
 - The script creates a bar plot to visualize the total wickets taken by each team. The sns.barplot function is used to create the plot, and the magma palette is used for coloring.  
  
11. Plot Win/Loss Ratio for Each Team:  
 plt.figure(figsize=(14, 8))  
 sns.barplot(x='win\_loss\_ratio', y='batting\_team', data=team\_performance, palette='coolwarm')  
 plt.title('Win/Loss Ratio for Each Team')  
 plt.xlabel('Win/Loss Ratio')  
 plt.ylabel('Team')  
 plt.show()  
 - The script creates a bar plot to visualize the win/loss ratio for each team. The sns.barplot function is used to create the plot, and the coolwarm palette is used for coloring.  
  
12. Plot Seasonal Performance:  
 plt.figure(figsize=(14, 8))  
 sns.lineplot(x='season', y='wins', hue='winner', data=seasonal\_performance, marker='o')  
 plt.title('Seasonal Performance of Teams')  
 plt.xlabel('Season')  
 plt.ylabel('Number of Wins')  
 plt.legend(title='Team', bbox\_to\_anchor=(1.05, 1), loc='upper left')  
 plt.xticks(rotation=45)  
 plt.tight\_layout()  
 plt.show()  
 - The script creates a line plot to visualize the seasonal performance of teams. The sns.lineplot function is used to create the plot, with markers for each data point. The legend is positioned outside the plot for better readability.